Secure Socket Layer Introduction and iKeyman

User's Guide

GSK Release 5C

Secure Socket Layer Introduction and iKeyman User's Guide (February 14, 2001)

Copyright Notice

Copyright © 2000 by Tivoli Systems Inc., an IBM Company, including this documentation and all software. All rights reserved. May only be used pursuant to a Tivoli Systems Software License Agreement or Addendum for Tivoli Products to IBM Customer or License Agreement. No part of this publication may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any computer language, in any form or by any means, electronic, mechanical, magnetic, optical, chemical manual, or otherwise, without prior written permission of Tivoli Systems. Tivoli Systems grants you limited permission to make hardcopy or other reproductions of any machine-readable documentation for your own use, provided that each such reproduction shall carry the Tivoli Systems copyright notice. No other rights under copyright are granted without prior written permission of Tivoli Systems. The document is not intended for production and is furnished "as is" without warranty of any kind. **All warranties on this document are hereby disclaimed including the warranties of merchantability and fitness for a particular purpose**.

Note to U.S. Government Users-Documentation related to restricted rights--Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corporation.

Trademarks

The following product names are trademarks of Tivoli Systems or IBM Corporation: AIX, IBM, OS/2, RS/6000, Tivoli, Tivoli ADSM, Tivoli Application Development Environment, Tivoli Application Extension Facility, Tivoli Asset Management, Tivoli Change Management, Tivoli Cross-Site for Availability, Tivoli Cross-Site for Deployment, Tivoli Cross-Site for Security, Tivoli Decision Support, Tivoli Developer Kit for PowerBuilder, Tivoli Distributed Monitoring, Tivoli Enterprise Console, Tivoli Event Integration Facility, Tivoli Global Enterprise Manager, Tivoli Integration Toolkit, Tivoli Inventory, Tivoli IT Director, Tivoli LAN Access, Tivoli Maestro, Tivoli Management Framework, Tivoli Manager for CATIA, Tivoli Manager for DB2, Tivoli Manager for Domino Tivoli Manager for Informix, Tivoli Manager for MCIS, Tivoli Manager for MCIS, Tivoli Manager for Microsoft Exchange, Tivoli Manager for MQSeries, Tivoli Manager for NIS SQL Server, Tivoli Manager for Oracle, Tivoli Manager for R/3, Tivoli Manager for SuiteSpot, Tivoli Manager for Sybase, Tivoli Manager for Year 2000, Tivoli Net Access, Tivoli NetView, Tivoli NetView Access Services, Tivoli NetView Distribution Manager, Tivoli NetView for OS/390, Tivoli NetView FTP, Tivoli NetView Performance Monitor, Tivoli OPC, Tivoli Output Manager, Tivoli Performance Reporter for OS/390, Tivoli Plus for ADSM, Tivoli Plus for AR System Tivoli Plus for Compaq Insight, Tivoli Problem Management, Tivoli Remote Control, Tivoli Reporter, Tivoli Security Management, Tivoli Service Desk, Tivoli Service Desk for OS/390, Tivoli Software Distribution, Tivoli User Administration, and Tivoli Workload Scheduler.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks or registered trademarks of Microsoft Corporation.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc., in the United States, other countries, or both.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

Other company, product, and service names mentioned in this document may be trademarks or servicemarks of others.

Notice

References in this publication to Tivoli Systems or IBM products, programs, or services do not imply that they will be available in all countries in which Tivoli Systems or IBM operates. Any reference to these products, programs, or services is not intended to imply that only Tivoli Systems or IBM products, programs, or services can be used. Subject to Tivoli Systems' or IBM's valid intellectual property or other legally protectable right, any functionally equivalent product, program, or service can be used instead of the referenced product, program or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by Tivoli Systems or IBM, are the responsibility of the user.

Tivoli Systems or IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, New York 10504-1785 U.S.A.

Contents

Figures vii
Preface ix

Chapter 1.Secure Sockets Layer Overview

Digital certificates 1
- Format of digital certificates 2
- Security considerations for digital certificates 3
- Certificate authorities and trust hierarchies 3
- Uses for digital certificates in Internet applications 5
- Digital certificates and certificate requests 6
- Global server certificates 7

How SSL works 7
- The SSL handshake 8
- Digital certificates and trust chains with SSL 9
- SSL with global server certificates 10

Chapter 2.Managing digital certificates with iKeyman

Creating a key database 11
Creating a self-signed digital certificate for testing 14
Adding a CA root digital certificate 15
Deleting a CA root digital certificate 15
Copying certificates from one key database to another 16
Requesting a digital certificate 18
Receiving a digital certificate 19
Deleting a digital certificate 20
Setting a new default key 20
Changing a database password 22
Managing digital certificates on a smart card 22

Chapter 3.Using the IKEYCMD Command Line Interface

Environment Set Up for IKEYCMD Command Line Interface 25
IKEYCMD Command Line Syntax 26
User Interface Task Reference 26
Creating a new key database 27
- Setting the database password 27
- Changing the database password 28
- Registering a key database with the server 28

Creating a new key pair and certificate request 28
Creating a self-signed certificate 29

Exporting keys 29
Importing keys 29
Listing CAs 30
Opening a key database 30
Receiving a CA-signed certificate 30
Showing the default key in a key database 31
Storing a CA certificate 31
Storing the encrypted database password in a stash file 31
Using GSK5CMD batch file 32
IKEYCMD Command Line Parameter Overview 32
IKEYCMD Command Line Options Overview 33
Command Line Invocation 34
User Properties File 36
Error Messages 36

Index

Figures

Figure 1. A simple directory tree 2
Figure 2. Simplified layout of a digital certificate 3
Figure 3. Chain of trust — CAs signing CA digital certificates up to the root CA 4
Figure 4. Self-signed digital certificate 6
Figure 5. SSL handshake with server authentication 8
Figure 6. New Key Database File window 11
Figure 7. Password Prompt window 12
Figure 8. IBM Key Management window 13
Figure 9. Create New Self-Signed Certificate window 14
Figure 10. Create New Key and Certificate Request window 19
Figure 11. Key Information window 21
Figure 12. Cryptographic Token in IBM Key Management Window 23
Figure 13. Open Cryptographic Token Window 24

Preface

About this book

This book is for system administrators who want to plan and integrate security for Web based systems. You should already understand your network and your e-business applications.

Who should read this book

This book is intended for network or system security administrators who install, administer, and use the Secure Socket Layer (SSL) based systems.

How this book is organized

This book contains the following chapters:

- "Chapter 1. Secure Sockets Layer Overview," on page 1 provides an overview of SSL and digital certificates.
- "Chapter 2. Managing digital certificates with iKeyman," on page 11 describes the iKeyman utility, which is a tool you can use to manage your digital certificates.

Conventions

This book uses the following conventions:

Convention	Meaning
bold	User interface elements such as check boxes, buttons, and commands
monospace	Syntax and directory defaults that are relevant to IBM SecureWay Toolkit
->	Shows a series of selections from a menu. For example: Select **File-> Run** means click **File**, and then click **Run**

1

Secure Sockets Layer Overview

Privacy and security are concepts that are more critical than ever in today's electronic business environment.

Every business professional needs to be concerned about security over open communication networks, such as the Internet. It is not enough to have a secure Web site; you also need to have secure communication *between* Web sites, communication that cannot be monitored by outside parties. Both you and your users need to be confident that you have a secure environment in which to conduct your business.

That kind of secure communication requires encryption, and encryption is what the Secure Sockets Layer (SSL) provides: security for the connection over which you can communicate.

SSL was developed jointly by Netscape Communications and RSA Data Security. Many companies worldwide have adopted SSL as their communication protocol of choice. In fact, many financial transactions on the Internet, including online banking, are now conducted using SSL.

Because digital certificates are an important component of SSL, this chapter consists of two sections:

- "Digital certificates"
- "How SSL works" on page 7

Digital certificates

Digital certificates allow unique identification of an entity; they are, in essence, electronic ID cards issued by trusted parties. Digital certificates allow a user to verify to whom a certificate is issued as well as the issuer of the certificate.

Digital certificates are the vehicle that SSL uses for public-key cryptography. Public-key cryptography uses two different cryptographic keys: a *private key* and a *public key*. Public-key cryptography is also known as *asymmetric cryptography*, because you can encrypt information with one key and decrypt it with the complement key from a given public-private *key pair*.

Public-private key pairs are simply long strings of data that act as keys to a user's encryption scheme. The user keeps the private key in a secure place (for example, encrypted on a computer's hard drive) and provides the public key to anyone with whom the user wants to communicate. The private key is used to digitally sign all secure communications sent from the user; the public key is used by the recipient to verify the sender's signature.

Public-key cryptography is built on trust; the recipient of a public key needs to have confidence that the key really belongs to the sender and not to an impostor. Digital certificates provide that confidence.

A digital certificate serves two purposes: it establishes the owner's identity, and it makes the owner's public key available. A digital certificate is issued by a trusted authority—a certificate authority (CA)—and it is issued only for a limited time. When its expiration date passes, the digital certificate must be replaced.

Format of digital certificates

The digital certificate contains specific pieces of information about the identity of the certificate owner and about the certificate authority:

- Owner's distinguished name. A distinguished name is the combination of the owner's common name and its context (position) in the directory tree. In the simple directory tree shown in Figure 1, for example, `LaurenA` is the owner's common name, and the context is `OU=Engnring.O=XYZCorp`; therefore, the distinguished name is:

  `.CN=LaurenA.OU=Engnring.O=XYZCorp`
- Owner's public key.
- Date the digital certificate was issued.
- Date the digital certificate expires.
- Issuer's distinguished name. This is the distinguished name of the CA.
- Issuer's digital signature.

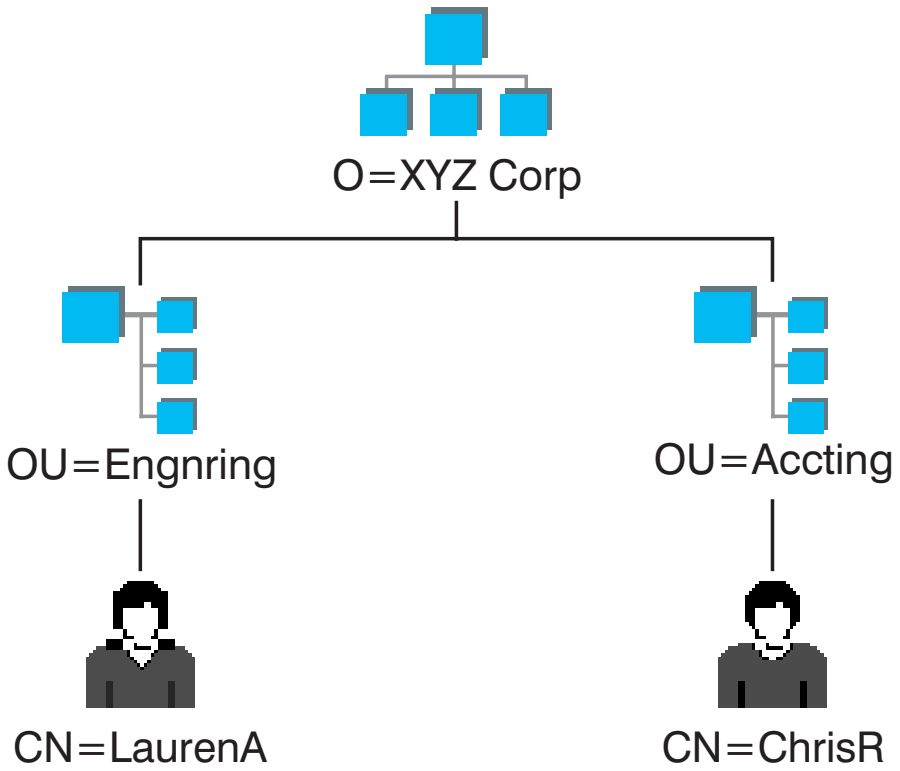

Figure 1. A simple directory tree

Figure 2 on page 3 shows the layout of a typical digital certificate.

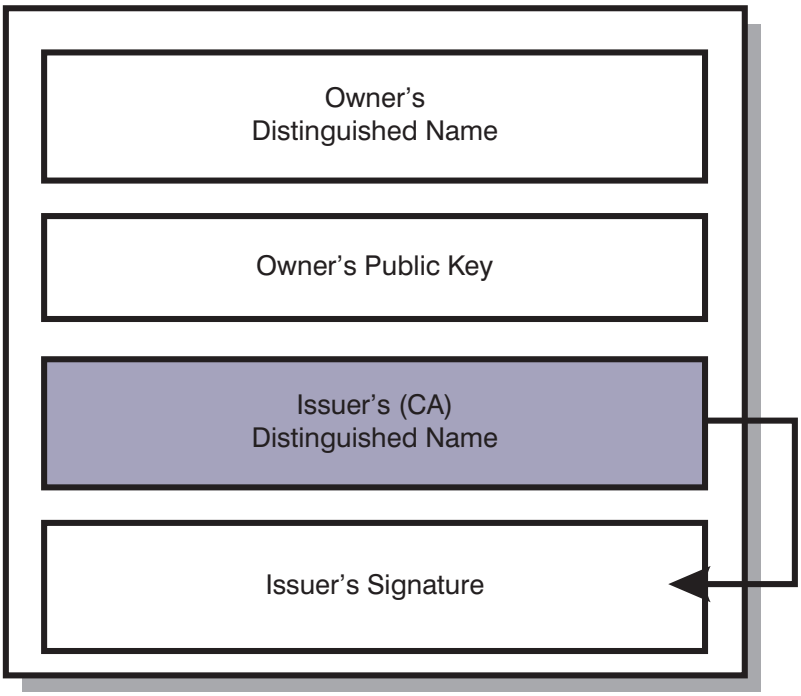

Figure 2. Simplified layout of a digital certificate

Security considerations for digital certificates

If you send your digital certificate containing your public key to someone else, what keeps that person from misusing your digital certificate and posing as you? The answer is *your private key*.

A digital certificate alone can never be proof of anyone's identity. The digital certificate just allows you to verify the identity of the digital certificate owner by providing the public key that is needed to check the digital certificate owner's digital signature. Therefore, the digital certificate owner must protect the private key that belongs to the public key in the digital certificate. If the private key is stolen, the thief can pose as the legitimate owner of the digital certificate. Without the private key, a digital certificate cannot be misused.

Certificate authorities and trust hierarchies

Trust is a very important concept in digital certificates. Each organization or user must determine which CAs can be accepted as trustworthy.

A user of a security service requiring knowledge of a public key generally needs to obtain and validate a digital certificate containing the required public key. Receiving a digital certificate from a remote party does not give the receiver any assurance about the authenticity of the digital certificate. To verify that the digital certificate is authentic, the receiver needs the public key of the certificate authority that issued the digital certificate.

If the public key user does not already hold an assured copy of the public key of the certificate authority that signed the digital certificate, then the user might need an additional digital certificate to obtain that public key. In general, a chain of multiple digital certificates might be needed, comprising a digital certificate of the public key owner (the end entity) signed by one CA, and optionally one or more additional digital certificates of CAs signed by other CAs. Figure 3 on page 4 shows a chain of trust.

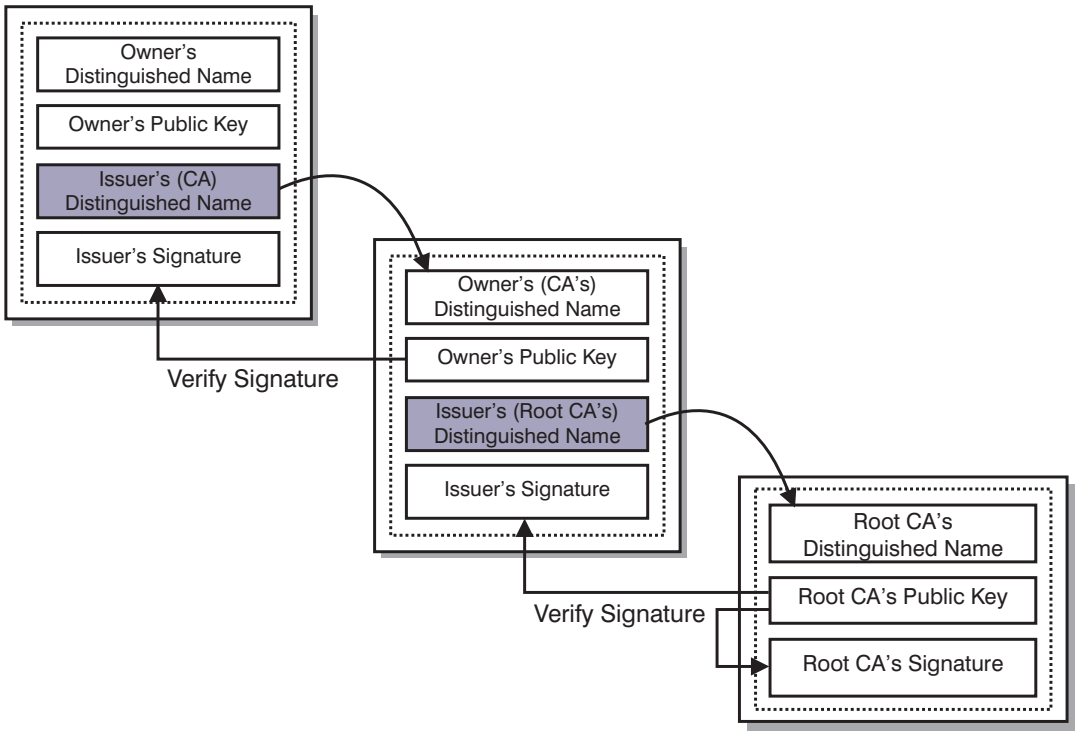

Figure 3. Chain of trust — CAs signing CA digital certificates up to the root CA

Note that many applications that send a subject's digital certificate to a receiver send not only that digital certificate, but also send all the CA digital certificates necessary to verify the initial digital certificate up to the root CA.

The chain of trust begins at the root CA. The root CA's digital certificate is self-signed; that is, the certificate authority uses its own private key to sign the digital certificate. The public key used to verify the signature is the public key in the digital certificate itself (see Figure 4 on page 6). To establish a chain of trust, the public-key user must have received the digital certificate of the root CA in one of the following ways:

- On a diskette received by registered mail or picked up in person
- Pre-loaded with software received from a reliable source or downloaded from an authenticated server

Uses for digital certificates in Internet applications

Applications using public-key cryptography systems for key exchange or digital signatures need to use digital certificates to obtain the needed public keys. Internet applications of this kind are numerous. Following are brief descriptions of a few of the commonly used Internet applications that use public-key cryptography:

(SSL) A protocol that provides privacy and integrity for communications. This protocol is used by Web servers to provide security for connections between Web servers and Web browsers, by the LDAP to provide security for connections between LDAP clients and LDAP servers, and by Host-on-Demand V2 to provide security for connections between the client and the host system. Additional applications based on this protocol are in development.

SSL uses digital certificates for key exchange, server authentication, and optionally, client authentication.

Client Authentication

Client authentication is an option in SSL that requires a server to authenticate a client's digital certificate before allowing the client to log on or access certain resources. The server requests and authenticates the client's digital certificate during the SSL handshake. At that time the server can also determine whether it trusts the CA that issued the digital certificate to the client.

Secure Electronic Mail

Many electronic mail systems, using standards such as *Privacy Enhanced Mail* (PEM) or *Secure/Multipurpose Internet Mail Extensions* (S/MIME) for secure electronic mail, use digital certificates for digital signatures and for the exchange of keys to encrypt and decrypt messages.

Virtual Private Networks (VPNs)

Virtual private networks, also called *secure tunnels*, can be set up between firewalls to enable protected connections between secure networks over insecure communication links. All traffic destined to these networks is encrypted between the firewalls.

The protocols used in tunneling follow the *IP Security* (IPsec) standard. For the key exchange between partner firewalls, the Internet key exchange (IKE) standard, previously known as ISAKMP/Oakley, has been defined.

The standards also allow for a secure, encrypted connection between a remote client (for example, an employee working from home) and a secure host or network.

Secure Electronic Transaction (SET)

SET is a standard designed for secure credit card payments in insecure networks (the Internet). Digital certificates are used for card holders (electronic credit cards) and merchants. The use of digital certificates in SET allows for secure, private connections between card holders, merchants, and banks. The transactions created are secure and indisputable, and they cannot be forged. The merchants receive no credit card information that can be misused or stolen.

Digital certificates and certificate requests

Simplified, a *signed* digital certificate contains the owner's distinguished name, the owner's public key, the certificate authority's (issuer's) distinguished name, and the signature of the certificate authority over these fields.

A *self-signed* digital certificate contains the owner's distinguished name, the owner's public key, and the owner's own signature over these fields, as shown in Figure 4.

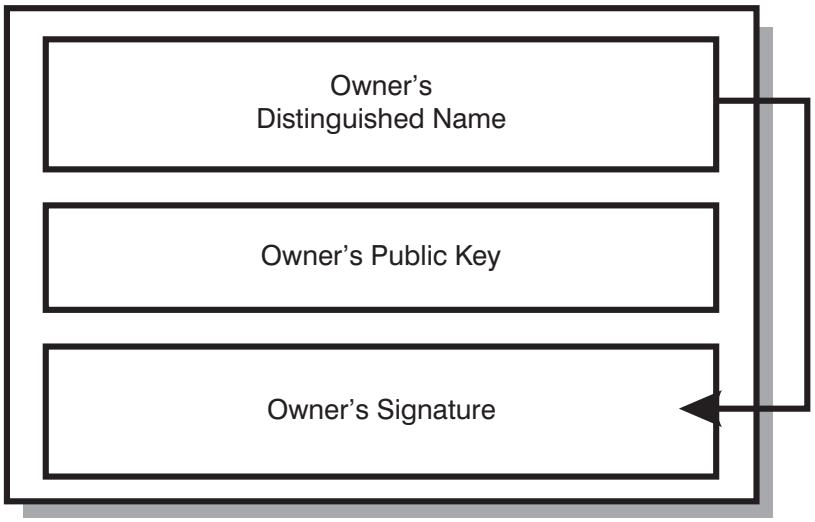

Figure 4. Self-signed digital certificate

A root CA's digital certificate is an example of a self-signed digital certificate. You can also create your own self-signed digital certificates to use when developing and testing a server product. See "Creating a self-signed digital certificate for testing" on page 14 for details.

A certificate request that is sent to a certificate authority to be signed contains the owner's (requester's) distinguished name, the owner's public key, and the owner's own signature over these fields. The certificate authority verifies this signature with the public key in the digital certificate to ensure that:

- The certificate request was not modified in transit between the requester and the CA.
- The requester is in possession of the private key that belongs to the public key in the certificate request.

The CA is also responsible for some level of identification verification. This can range from very little proof to absolute assurance of the owner's identity.

Global server certificates

There is a special kind of server digital certificate for Web servers, called a *global server certificate*.

Global server certificates are needed because of restrictions on export of cryptographic hardware and software imposed by the United States government:

- Users in the United States and Canada can use any available cryptographic algorithm with any key length. Delivery of cryptographic hardware and software to customers in the United States and Canada is unrestricted.
- Users in other countries using United States products may use only cryptographic algorithms up to certain key lengths. Delivery of cryptographic hardware and software to customers outside the United States and Canada is restricted and controlled by the United States government.

The United States government export regulations allow certain industries in countries outside the United States and Canada (currently financial institutions such as banks, insurance companies, and health industry organizations) to use cryptographic products with the same key lengths as in the United States.

To comply with the U.S. government export regulations, global server certificates were created. The United States government has authorized VeriSign, Inc. to issue special server digital certificates to customers eligible to use strong encryption, such as banks and other financial institutions, insurance companies, and health-industry organizations. These digital certificates are recognized by Microsoft Internet Explorer Version 3.02 and later and by Netscape Navigator/Communicator Version 4 and later. When the Web browser recognizes the special digital certificate, it enables strong encryption routines, such as RC4 with 128-bit keys or Triple DES with 168-bit keys. The SSL Toolkit supports global server certificates in the same fashion.

Global server certificates can be used by:

- Banks, financial institutions, insurance companies, and healthcare organizations outside the United States and Canada that require SSL between their Web servers and their customers' and users' Web browsers with encryption stronger than RC2 or RC4 with 40-bit keys.
- Banks, financial institutions, insurance companies, and healthcare organizations inside the United States or Canada that require SSL between their Web servers and the Web browsers of customers or users located outside the United States or Canada with encryption stronger than RC2 or RC4 with 40-bit keys.

How SSL works

SSL is a protocol that provides privacy and integrity between two communicating applications using TCP/IP. The *Hypertext Transfer Protocol* (HTTP) for the World Wide Web uses SSL for secure communications.

The data going back and forth between client and server is encrypted using a symmetric algorithm such as DES or RC4. A public-key algorithm—usually RSA—is used for the exchange of the encryption keys and for digital signatures. The algorithm uses the public key in the server's digital certificate. With the server's digital certificate, the client can also verify the server's identity. Versions 1 and 2 of the SSL protocol provide only server authentication. Version 3 adds client authentication, using both client and server digital certificates.

The SSL handshake

A (SSL) connection is always initiated by the client using a URL starting with `https://` instead of with `http://`. At the beginning of an SSL session, an SSL handshake is performed. This handshake produces the cryptographic parameters of the session. A simplified overview of how the SSL handshake is processed is shown in Figure 5.

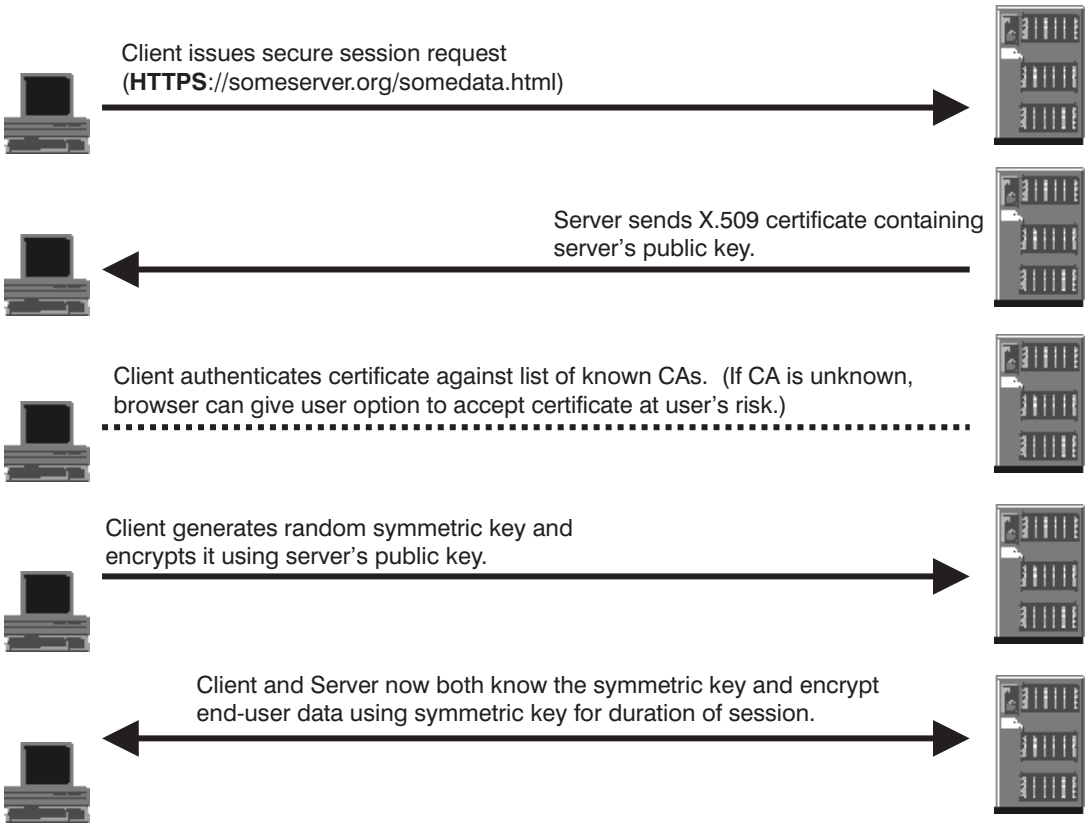
Client issues secure session request
(**HTTPS**://someserver.org/somedata.html)
Server sends X.509 certificate containing
server's public key.
Client authenticates certificate against list of known CAs. (If CA is unknown,
browser can give user option to accept certificate at user's risk.)
Client generates random symmetric key and
encrypts it using server's public key.
Client and Server now both know the symmetric key and encrypt
end-user data using symmetric key for duration of session.

Figure 5. SSL handshake with server authentication

1. The client sends a client "hello" message that lists the cryptographic capabilities of the client (sorted in client preference order), such as the version of SSL, the cipher suites supported by the client, and the data compression methods supported by the client. The message also contains a 28-byte random number.
2. The server responds with a server "hello" message that contains the cryptographic method (cipher suite) and the data compression method selected by the server, the session ID, and another random number.

   Note: The client and the server must support at least one common cipher suite, or else the handshake fails. The server generally chooses the strongest common cipher suite.

3. The server sends its digital certificate. (The server uses X.509 V3 digital certificates with SSL.)

   If the server uses SSL V3, and if the server application (for example, the Web server) requires a digital certificate for client authentication, the server sends a "digital certificate request" message. In the "digital certificate request" message, the server sends a list of the types of digital certificates supported and the distinguished names of acceptable certificate authorities.

4. The server sends a server "hello done" message and waits for a client response.

5. Upon receipt of the server "hello done" message, the client (the Web browser) verifies the validity of the server's digital certificate and checks that the server's "hello" parameters are acceptable.

   If the server requested a client digital certificate, the client sends a digital certificate, or if no suitable digital certificate is available, the client sends a "no digital certificate" alert. This alert is only a warning, but the server application can fail the session if client authentication is mandatory.

6. The client sends a "client key exchange" message. This message contains the *pre-master secret*, a 46-byte random number used in the generation of the symmetric encryption keys and the *message authentication code* (MAC) keys, encrypted with the public key of the server.

   If the client sent a digital certificate to the server, the client sends a "digital certificate verify" message signed with the client's private key. By verifying the signature of this message, the server can explicitly verify the ownership of the client digital certificate.

   Note: An additional process to verify the server digital certificate is not necessary. If the server does not have the private key that belongs to the digital certificate, it cannot decrypt the pre-master secret and create the correct keys for the symmetric encryption algorithm, and the handshake fails.

7. The client uses a series of cryptographic operations to convert the pre-master secret into a *master secret*, from which all key material required for encryption and message authentication is derived. Then the client sends a "change cipher spec" message to make the server switch to the newly negotiated cipher suite. The next message sent by the client (the "finished" message) is the first message encrypted with this cipher method and keys.

8. The server responds with a "change cipher spec" and a "finished" message of its own.

9. The SSL handshake ends, and encrypted application data can be sent.

Digital certificates and trust chains with SSL

Secure Sockets Layer V3 can use server digital certificates as well as client digital certificates. As previously explained, server digital certificates are mandatory for an SSL session, while client digital certificates are optional, depending on client authentication requirements.

The *public key infrastructure* (PKI) used by SSL allows for any number of root certificate authorities. An organization or end user must decide for itself which CAs it will accept as being *trusted*. To be able to verify the server digital certificates, client Web browsers require possession of the root CA digital certificates used by servers. Popular Web browsers such as Netscape Navigator/Communicator or Microsoft Internet Explorer usually come with a *key ring* where a number of CA digital certificates, called trusted roots, are already installed. This list can be edited, and the digital certificates of untrusted CAs can be deleted.

If an SSL session is about to be established with a server that sends a digital certificate whose root CA digital certificate is not in the key ring, the browser displays a warning window and presents options either to import the digital certificate or to abort the session. To avoid this situation, import the root CA digital certificate from a Web page or use a JavaScript program that imports the digital certificate.

If client authentication is used, the Web server requires possession of the root CA digital certificates used by clients. Because it is not possible to import root CA digital certificates into the server application dynamically, all root CA digital certificates that are not part of the server key ring at delivery time must be installed using the iKeyman utility before any client digital certificates are issued by these CAs. For more information on iKeyman, see “Chapter 2. Managing digital certificates with iKeyman,” on page 11.

SSL with global server certificates

When an SSL session is established between the international version of Netscape Navigator/Communicator V4, Microsoft Internet Explorer V4, IBM SSL-enabled client applications, or client applications written using the SSL Toolkit and a Web server equipped with a global server certificate, a normal SSL handshake (described in Figure 5 on page 8) is performed initially. Usually, the Web server and the Web browser settle on the cipher suite **SSL_RSA_EXPORT_WITH_RC4_40_MD5**. They use 512-bit RSA keys for key exchange, RC4 with 40-bit keys for encryption, and MD5 for message authentication.

During the handshake, the browser receives and verifies the server digital certificate and realizes that this digital certificate authorizes stronger encryption. Remember that the browser sends the list of cryptographic suites it supports with the "client hello" message before the server sends its digital certificate. At this point, the browser has no knowledge of the server's global server certificate.

The first handshake is completed with the "change cipher specification" and "finished" messages from both client and server. At this point, the client initiates another SSL handshake. This time, in the "client hello" message, the client includes the strong cryptographic suites such as:

- **SSL_RSA_WITH_RC4_128_MD5** (1024-bit RSA keys for key exchange, RC4 with 128-bit keys for encryption, and MD5 for message authentication)
- **SSL_RSA_WITH_3DES_EDE_CBC_SHA** (1024-bit RSA keys for key exchange, triple DES with 168-bit keys for encryption, and SHA-1 for message authentication)

After the second handshake is completed, one of the stronger cryptographic suites is used.

This double handshake is also known as the *SSL step-up protocol*. Actually, all application data exchanged in the SSL session are encrypted with the stronger encryption protocol. Compared to the use of a United States browser, the only drawback is the higher overhead of performing an SSL handshake twice.

2

Managing digital certificates with iKeyman

The iKeyman utility is a tool you can use to manage your digital certificates. With iKeyman, you can create a new key database or a test digital certificate, add CA roots to your database, copy certificates from one database to another, request and receive a digital certificate from a CA, set default keys, and change passwords. You can also use the iKeyman utility to perform many of these same functions for digital certificates on a smart card.

The iKeyman utility is automatically installed with the SSL Toolkit.

Creating a key database

A key database enables a client application to connect to those servers that have digital certificates signed by those CAs for which you have signed digital certificates.

To create a CMS key database file, follow these steps:

1. Start iKeyman (for example, click **Start→ Programs→ GSKit→ Ikeyman Gui Shortcut**). The **IBM Key Management** window is displayed.
2. Click **Key Database File→ New**. The **New** window is displayed, as shown in Figure 6.

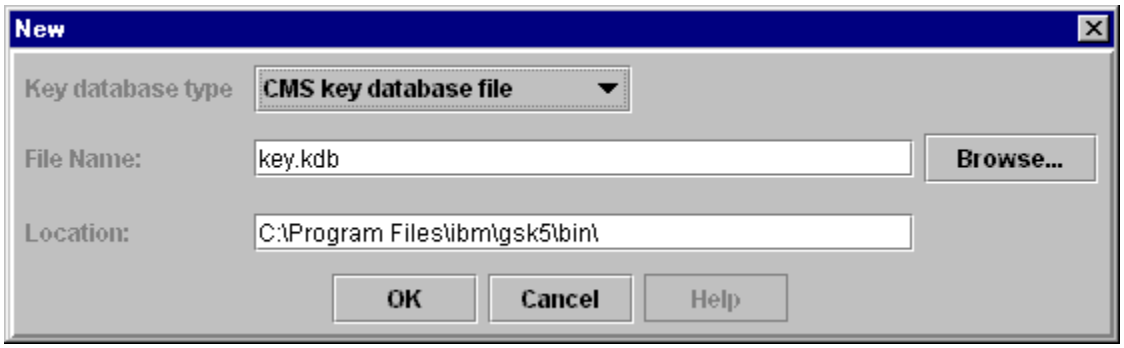

Figure 6. New Key Database File window

3. Select **CMS key database file** for the **Key database type** field.
4. Type a **File Name**, such as `key.kdb`.
5. Accept the default value for the **Location** field, or type a new value for the field, or use the **Browse** button to select a new value.
6. Click **OK**. The **Password Prompt** window is displayed, as shown in Figure 7 on page 12.

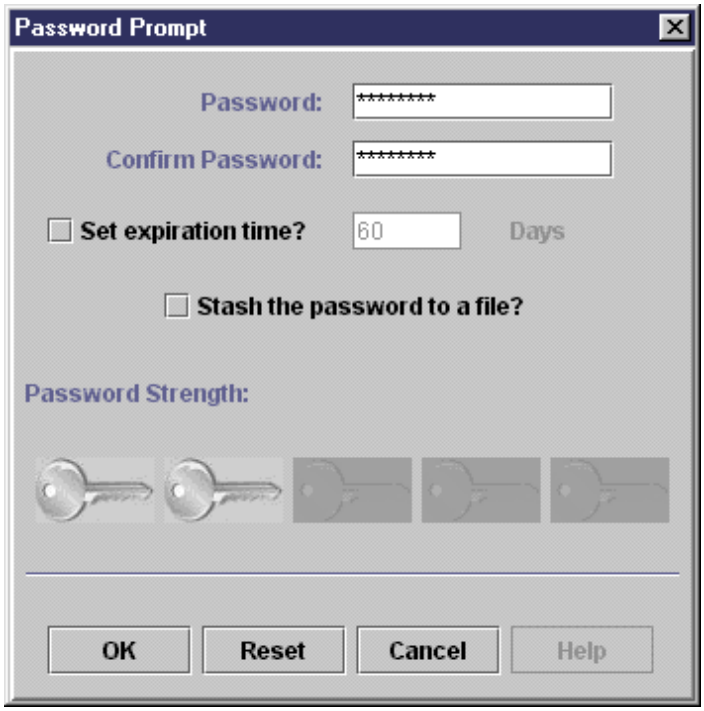

Figure 7. Password Prompt window

7. Type a password in the **Password** field, and type it again in the **Confirm Password** field.
8. Click **OK**. A confirmation window is displayed, verifying that you have created a key database.
9. Click **OK**. You have successfully created a key database, and the **IBM Key Management** window is displayed.

As shown in Figure 8 on page 13, the **IBM Key Management** window now reflects your new CMS key database file (for example, `C:\Program Files\ibm\gsk5\bin\key.kdb`), and your signer digital certificates.

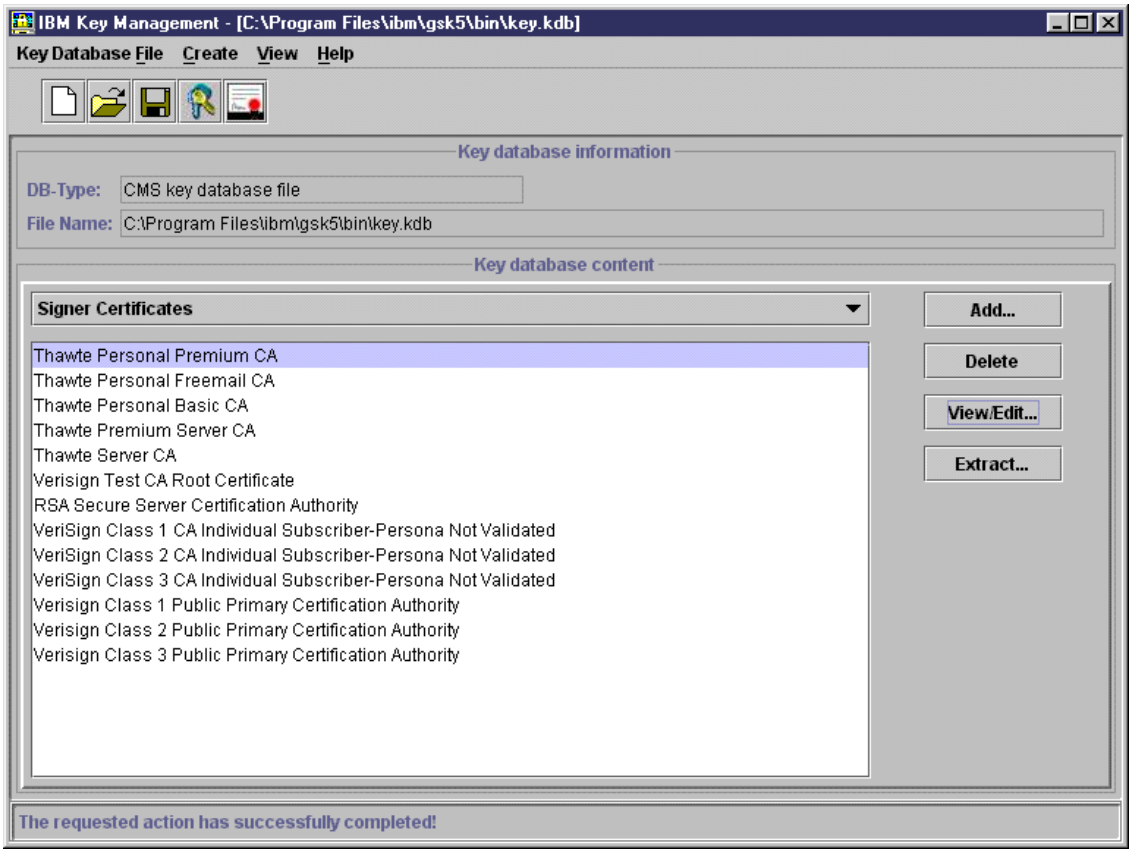

Figure 8. IBM Key Management window

The following signer digital certificates are provided with iKeyman:

- RSA Secure Server Certification Authority
- Thawte Personal Basic CA
- Thawte Personal Freemail CA
- Thawte Personal Premium CA
- Thawte Premium Server CA
- Thawte Server CA
- VeriSign Class 1 CA Individual-Persona Not Validated
- VeriSign Class 2 CA Individual-Persona Not Validated
- VeriSign Class 3 CA Individual-Persona Not Validated
- VeriSign Class 1 Public Primary Certification Authority
- VeriSign Class 2 Public Primary Certification Authority
- VeriSign Class 3 Public Primary Certification Authority
- VeriSign Test CA Root Certificate

These signer digital certificates enable your clients to connect to servers that have valid digital certificates from these signers. Now that you have created a key database, you can use it on your client and connect to a server that has a valid digital certificate from one of the signers.

If you need to use a signer digital certificate that is not on this list, you need to request it from the CA and add it to your key database (see “Adding a CA root digital certificate” on page 15).

Note: The *VeriSign Test CA Root Certificate* is a low-assurance CA that is included for testing purposes. You should remove this root before placing a key database class into a production application.

Creating a self-signed digital certificate for testing

When you are developing a production application, you might not want to purchase a true digital certificate until after you are done testing the product. With iKeyman, you can create a self-signed digital certificate to use until testing is complete. A self-signed digital certificate is a temporary digital certificate you issue to yourself, with yourself as the CA.

Note: Do not release a production application with a self-signed digital certificate; no browser or client will be able to recognize or communicate with your server.

To create a self-signed digital certificate in a key database, follow these steps:

1. Start iKeyman (for example, click **Start→ Programs→ GSKit→ Ikeyman Gui Shortcut).** The **IBM Key Management** window is displayed.
2. Click **Key Database File→ Open**. The **Open** window is displayed.
3. Select the key database file to which you want to add a self-signed digital certificate and click **Open**. The **Password Prompt** window is displayed.
4. Type the password and click **OK**. The **IBM Key Management** window is displayed. The title bar shows the name of the key database file you selected, indicating that the file is open and ready.
5. Select **Personal Certificates** from the pulldown list.
6. Click **New Self-Signed**. The **Create New Self-Signed Certificate** window is displayed, as shown in Figure 9.

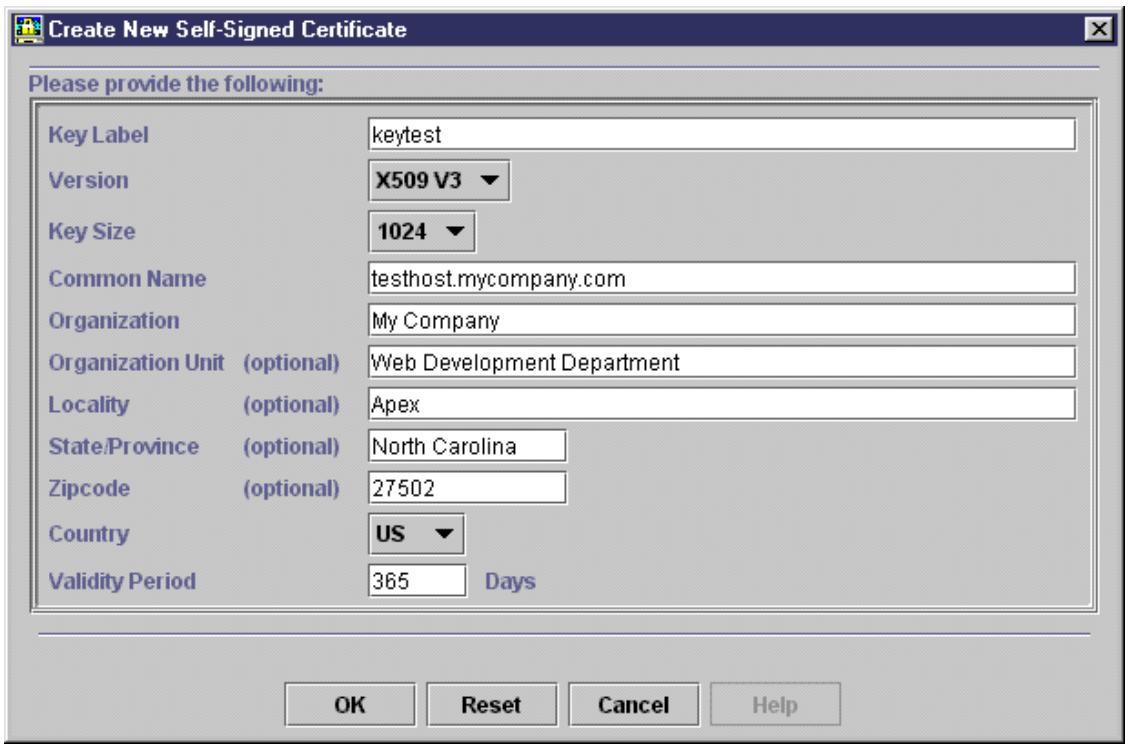

Figure 9. Create New Self-Signed Certificate window

7. Type a **Key Label**, such as `keytest`, for the self-signed digital certificate.
8. Type a **Common Name** and **Organization**, and select a **Country**. For the remaining fields, either accept the default values, or type or select new values.
9. Click **OK**. The **IBM Key Management** window is displayed. The **Personal Certificates** field shows the name of the self-signed digital certificate you created.

Adding a CA root digital certificate

After you have requested and received a CA root digital certificate from a CA, you can add it to your database. Most root digital certificates have the form *.arm (for example, `cert.arm`).

To add a CA root digital certificate to a database, follow these steps:

1. Start iKeyman (for example, click **Start→ Programs→ GSKit→ Ikeyman Gui Shortcut).** The **IBM Key Management** window is displayed.
2. Click **Key Database File→ Open**. The **Open** window is displayed.
3. Select the key database file to which you want to add a CA root digital certificate and click **Open**. The **Password Prompt** window is displayed.
4. Type the password and click **OK**. The **IBM Key Management** window is displayed. The title bar shows the name of the key database file you selected, indicating that the file is open and ready.
5. Select **Signer Certificates** from the pulldown list.
6. Click **Add**. The **Add CA's Certificate from a File** window is displayed.
7. Click **Data type** and select a data type, such as **Base64-encoded ASCII data**.
8. Type a **Certificate file name** and **Location** for the CA root digital certificate, or click **Browse** to select the name and location.
9. Click **OK**. The **Enter a Label** window is displayed.
10. Type a label for the CA root digital certificate, such as `VeriSign Test CA Root Certificate`, and click **OK**. The **IBM Key Management** window is displayed.

    The **Signer Certificates** field now shows the label of the CA root digital certificate you just added.

Deleting a CA root digital certificate

If you no longer want to support one of the signers in your signer digital certificate list, you need to delete the CA root digital certificate.

Note: Before deleting a CA root digital certificate, create a backup copy in case you later want to re-create the CA root.

To delete a CA root digital certificate from a database, follow these steps:

1. Start iKeyman (for example, click **Start→ Programs→ GSKit→ Ikeyman Gui Shortcut).** The **IBM Key Management** window is displayed.
2. Click **Key Database File→ Open**. The **Open** window is displayed.
3. Select the key database file from which you want to delete a CA root digital certificate and click **Open**. The **Password Prompt** window is displayed.

4. Type the password and click **OK**. The **IBM Key Management** window is displayed. The title bar shows the name of the key database file you selected, indicating that the file is open and ready.
5. Select **Signer Certificates** from the pulldown list.
6. Select the CA root digital certificate you want to delete and click **Delete**. The **Confirm** window is displayed.
7. Click **Yes**. The **IBM Key Management** window is displayed. The label of the CA root digital certificate you just deleted no longer appears in the **Signer Certificates** field.

Copying certificates from one key database to another

When setting up a private trust network or using self-signed certificates for testing purposes, you might find it necessary to extract a certificate from a database to be added to another database as a signer certificate. Other times, you might want to export a personal certificate and import it as a personal certificate.

First scenario: To extract a certificate from the (source) key database to be added as a signer certificate in the (target) key database, follow these steps:

1. Start iKeyman (for example, click **Start→ Programs→ GSKit→ Ikeyman Gui Shortcut).** The **IBM Key Management** window is displayed.
2. Click **Key Database File→ Open**. The **Open** window is displayed.
3. Select the (source) key database containing the certificate that you would like to add to another (target) database as a signer certificate and click **Open**. The **Password Prompt** window is displayed.
4. Type the password and click **OK**. The **IBM Key Management** window is displayed. The title bar shows the name of the key database file you selected, indicating that the class is open and ready.
5. Select the type of certificate you want to export: **Personal**, or **Signer**.
6. Select the certificate that you want to add to another database.
7. If you select **Personal,** click the **Extract Certificate** pushbutton. If you select **Signer** click the **Extract** pushbutton. The **Extract a Certificate to a File** window is displayed. You will proceed with the remaining steps.
8. Click **Data type** and select a data type, such as **Base64-encoded ASCII data**. The data type needs to match the data type of the certificate stored in the certificate file. The iKeyman tool supports Base64-encoded ASCII files and binary DER-encoded certificates.
9. Type the certificate file name and location where you want to store the certificate, or click **Browse** to select the name and location.
10. Click **OK**. The certificate is written to the specified file, and the **IBM Key Management** window is displayed.

To add a certificate as a signer certificate to the database (target), follow these steps:

1. Start iKeyman (for example, click **Start→ Programs→ GSKit→ Ikeyman Gui Shortcut).** The **IBM Key Management** window is displayed.
2. Click **Key Database File→ Open**. The **Open** window is displayed.
3. Select the key database to which you would like to add the certificate that has been extracted from above and click **Open**. The **Password Prompt** window is displayed.

4. Type the password and click **OK**. The **IBM Key Management** window is displayed. The title bar shows the name of the key database file you selected, indicating that the class is open and ready.
5. Select the type of certificate you would like to add: **Signer**.
6. Click **Add**. The **Add CA's Certificate from a File** window displays.
7. Type the certificate file name that you used when you extracted a certificate. For more information, see step 9 on page 16 above.
8. The **Enter a Label** window displays.
9. Specify the name of the certificate, and click **OK**.

   The certificate is now added to the (target) database.

Second scenario: In the previous scenario, you extracted a personal, or signer certificate from a source database and added it to the target database as a signer certificate. This scenario exports a personal certificate from a source database and imports it to a target database as a personal certificate.

To export a personal certificate from the (source) key database to be imported as a personal certificate in the (target) key database follow these steps:

1. Start iKeyman (for example, click **Start→ Programs→ GSKit→ Ikeyman Gui Shortcut).** The **IBM Key Management** window is displayed.
2. Click **Key Database File→ Open**. The **Open** window is displayed.
3. Select the (source) key database containing the certificate that you would like to add to another (target) key database as a personal certificate and click **Open**. The **Password Prompt** window is displayed.
4. Type the password and click **OK**. The **IBM Key Management** window is displayed. The title bar shows the name of the key database file you selected, indicating that the class is open and ready.
5. Select **Personal Certificates** from the pulldown list.
6. Select the personal certificate you want to export.
7. Select the **Export/Import** pushbutton to transfer keys between the current database and a PKCS#12 file or another database. The **Export/Import Key** window displays.
8. Select **Export** from the Choose Action Type.
9. Select **Key File Type** (for example, PKCS12 file) from the pulldown to export list.
10. Type the name of the file (for example, `copy.p12`) to which you would like to export the certificate, or click **Browse** to select the name and location, and click **OK**. The **Password Prompt** window displays.
11. Enter a password for the certificate file, confirm the password, and click **OK.**

    The certificate is now exported from the (source) database.

    Note: The PKCS#12 file is a temporary file and should be deleted after use.

To import a personal certificate to the (target) key database, follow these steps:

1. Start iKeyman (for example, click **Start→ Programs→ GSKit→ Ikeyman Gui Shortcut).** The **IBM Key Management** window is displayed.
2. Click **Key Database File→ Open**. The **Open** window is displayed.

3. Select the (target) key database to which you would like to import the certificate that has been exported above and click **Open**. The **Password Prompt** window is displayed.
4. Type the password and click **OK**. The **IBM Key Management** window is displayed. The title bar shows the name of the key database file you selected, indicating that the class is open and ready.
5. Select the **Personal Certificates** from the pulldown list.
6. If the target key database has no personal certificate, click the **Import** pushbutton to import keys from a PKCS#12 file or another database. The **Import Key** window displays. If target key database has one or more personal certificates, do the following:
   - Click the **Export/Import key** pushbutton, the **Export/Import key** window displays.
   - Select **Import** from the Choose Action Type.
7. Select the same key file type that you specified from the export. For more information, see step 9 on page 16.
8. Type the name of the file containing the certificate you exported, or click **Browse** to select the name and location. For more information, see step 10 on page 17. Click **OK**. The **Password Prompt** window displays.
9. Specify the password from when you exported the certificate. For more information, see step 11 on page 17. Click **OK**.

   The certificate is now imported to the (target) database.

Requesting a digital certificate

A digital certificate is required to run the SSL-enabled server code and might be required for client applications. To acquire a digital certificate, generate a request using iKeyman and submit the request to a CA. The CA will verify your identity and send you a digital certificate.

To request a digital certificate for a key database, follow these steps:

1. Start iKeyman (for example, click **Start→ Programs→ GSKit→ Ikeyman Gui Shortcut).** The **IBM Key Management** window is displayed.
2. Click **Key Database File→ Open**. The **Open** window is displayed.
3. Select the key database file from which you want to generate the request and click **Open**. The **Password Prompt** window is displayed.
4. Type the password and click **OK**. The **IBM Key Management** window is displayed. The title bar shows the name of the key database file you selected, indicating that the file is open and ready.
5. Select **Personal Certificate Requests** from the pulldown list.
6. Click **New**. The **Create New Key and Certificate Request** window is displayed, as shown in Figure 10 on page 19.

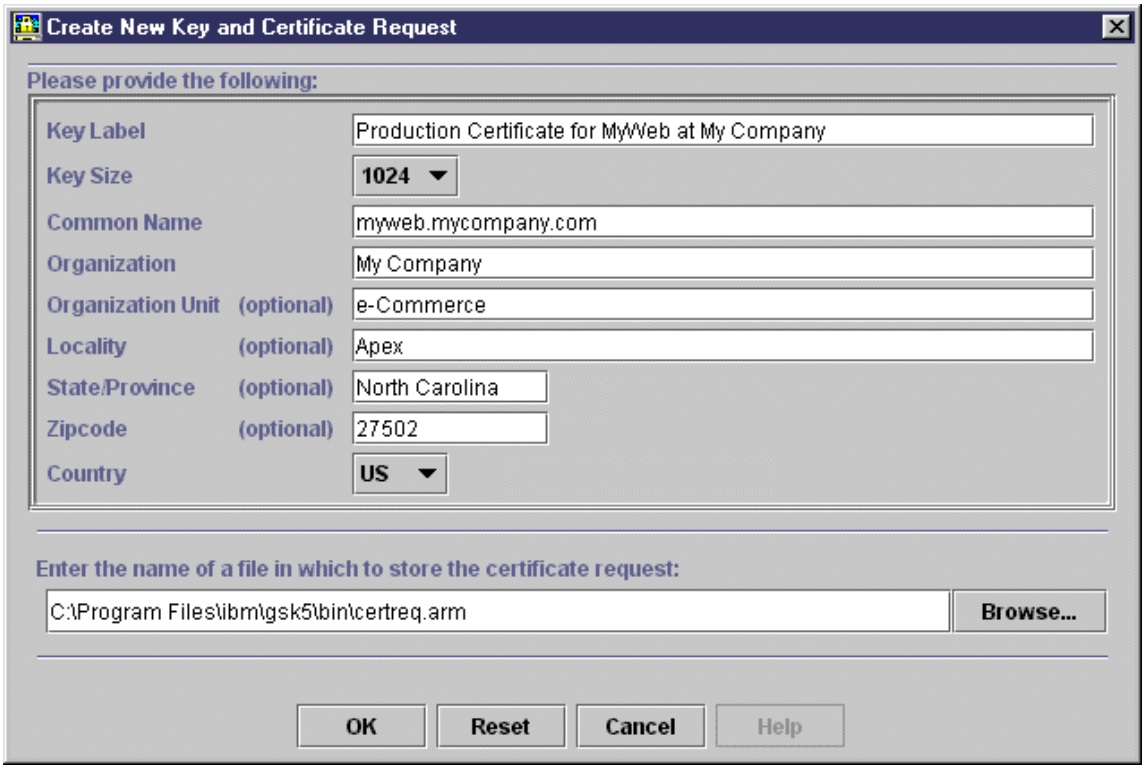

Figure 10. Create New Key and Certificate Request window

7. Type a **Key Label**, such as `Production Certificate for MyWeb at My Company`, for the digital certificate request.
8. Type a **Common Name** and **Organization**, and select a **Country**. For the remaining fields, either accept the default values, or type or select new values.
9. At the bottom of the window, type a name for the file, such as `certreq.arm`.
10. Click **OK**. A confirmation window is displayed, verifying that you have created a request for a new digital certificate.
11. Click **OK**. The **IBM Key Management** window is displayed. The **Personal Certificate Requests** field shows the key label of the new digital certificate request you created.
12. Send the file to a CA to request a new digital certificate, or cut and paste the request into the request forms of the CA's Web site.

Receiving a digital certificate

After the CA sends you a new digital certificate, you need to add it to the key database from which you generated the request.

To receive a digital certificate into a key database, follow these steps:

1. Start iKeyman (for example, click **Start→ Programs→ GSKit→ Ikeyman Gui Shortcut).** The **IBM Key Management** window is displayed.
2. Click **Key Database File→ Open**. The **Open** window is displayed.
3. Select the key database file from which you generated the request and click **Open**. The **Password Prompt** window is displayed.

4. Type the password and click **OK**. The **IBM Key Management** window is displayed. The title bar shows the name of the key database file you selected, indicating that the file is open and ready.
5. Select **Personal Certificates** from the pulldown list.
6. Click **Receive**. The **Receive Certificate from a File** window is displayed.
7. Click **Data type** and select the data type of the new digital certificate, such as **Base64-encoded ASCII data**. If the CA sends the certificate as part of an e-mail message, then you might need to cut and paste the certificate into a separate file.
8. Type the **Certificate file name** and **Location** for the new digital certificate, or click **Browse** to select the name and location.
9. Click **OK**. The **Enter a Label** window is displayed.
10. Type a label, such as `Production Certificate for MyWeb at My Company`, for the new digital certificate and click **OK**. The **IBM Key Management** window is displayed. The **Personal Certificates** field shows the label of the new digital certificate you added.

Deleting a digital certificate

If you no longer need one of your digital certificates, you need to delete it from your database.

Note: Before deleting a digital certificate, create a backup copy in case you later want to re-create it.

To delete a digital cerificate from a key database, follow these steps:

1. Start iKeyman (for example, click **Start→ Programs→ GSKit→ Ikeyman Gui Shortcut).** The **IBM Key Management** window is displayed.
2. Click **Key Database File→ Open**. The **Open** window is displayed.
3. Select the key database file from which you want to delete the digital certificate and click **Open**. The **Password Prompt** window is displayed.
4. Type the password and click **OK**. The **IBM Key Management** window is displayed. The title bar shows the name of the key database file you selected, indicating that the file is open and ready.
5. Select **Personal Certificates** from the pulldown list.
6. Select the digital certificate you want to delete and click **Delete**. The **Confirm** window is displayed.
7. Click **Yes**. The **IBM Key Management** window is displayed. The label of the digital certificate you just deleted no longer appears in the **Personal Certificates** field.

Setting a new default key

To make it easy to configure an SSL application, the iKeyman utility lets you specify a default digital certificate for applications to use when the key database contains more than one Personal Certificate entry. (You might have more than one digital certificate in a database if you started using a self-signed digital certificate in the application (for testing) while waiting for the official digital certificate from your chosen CA.) After receiving the official digital certificate from the CA, you can leave the self-signed digital certificate in the database and begin using the CA-issued digital certificate by making it the default digital certificate. The default digital certificate is indicated by an asterisk (*) in front of the entry label.

The first digital certificate that is received or created as self-signed is marked as the default digital certificate. Each time a new digital certificate is received or a self-signed digital certificate is created, you are given the option to make that digital certificate the default digital certificate. However, you can also explicitly change the default digital certificate at any time.

To change the default digital certificate in a key database, follow these steps:

1. Start iKeyman (for example, click **Start→ Programs→ GSKit→ Ikeyman Gui Shortcut).** The **IBM Key Management** window is displayed.
2. Click **Key Database File→ Open**. The **Open** window is displayed.
3. Select the key database file in which you want to change the default digital certificate and click **Open**. The **Password Prompt** window is displayed.
4. Type the password and click **OK**. The **IBM Key Management** window is displayed. The title bar shows the name of the key database file you selected, indicating that the file is open and ready.
5. Select **Personal Certificates** from the pulldown list. The default digital certificate is indicated by an asterisk (*) in front of the entry label.
6. Select the digital certificate you want to set as the default digital certificate and click **View/Edit**, or double-click on the entry. The **Key Information** window is displayed for the digital certificate entry, as shown in Figure 11.

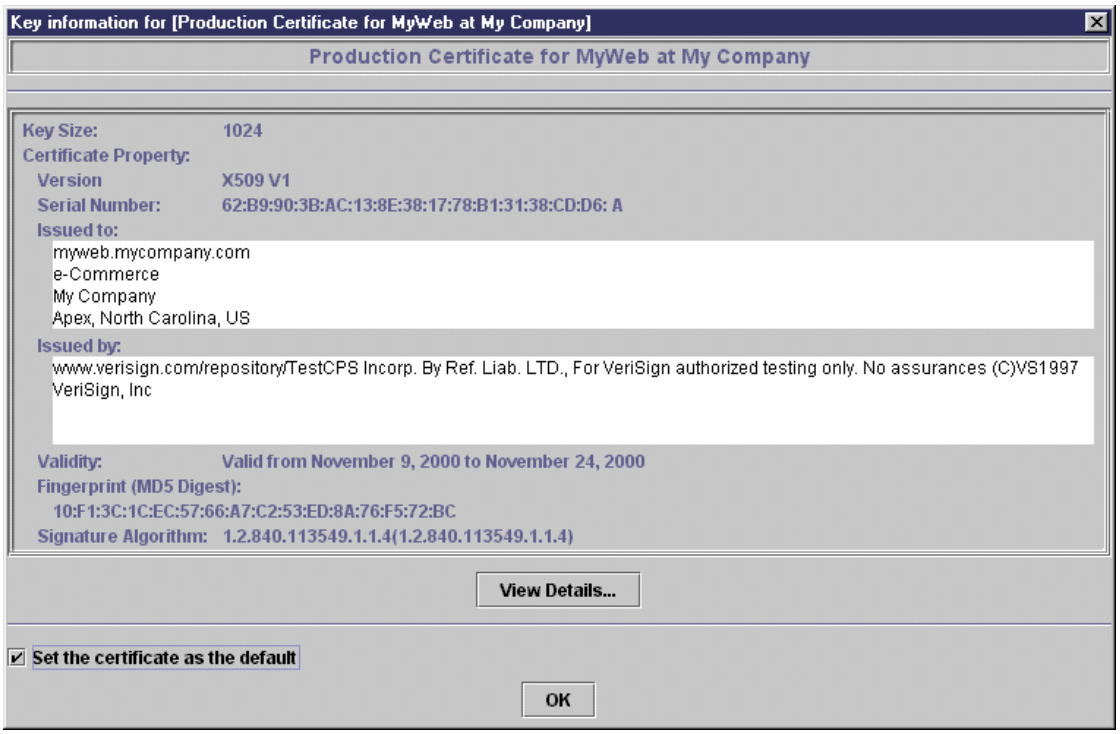
Key information for [Production Certificate for MyWeb at My Company]
Production Certificate for MyWeb at My Company
Key Size: 1024
Certificate Property:
Version X509 V1
Serial Number: 62:B9:90:3B:AC:13:8E:38:17:78:B1:31:38:CD:D6: A
Issued to:
myweb.mycompany.com
e-Commerce
My Company
Apex, North Carolina, US
Issued by:
www.verisign.com/repository/TestCPS Incorp. By Ref. Liab. LTD., For VeriSign authorized testing only. No assurances (C)VS1997
VeriSign, Inc
Validity: Valid from November 9, 2000 to November 24, 2000
Fingerprint (MD5 Digest):
10:F1:3C:1C:EC:57:66:A7:C2:53:ED:8A:76:F5:72:BC
Signature Algorithm: 1.2.840.113549.1.1.4(1.2.840.113549.1.1.4)
View Details...
Set the certificate as the default
OK

Figure 11. Key Information window

7. To set this digital certificate as the default digital certificate, select **Set the certificate as the default** and click **OK**. The **IBM Key Management** window is displayed. The label of the digital certificate you just set as the default is identified as the default digital certificate by an asterisk (*) in front of the entry.

Changing a database password

The iKeyman tool allows you to change a database password.

To change a key database password, follow these steps:

1. Start iKeyman (for example, click **Start→ Programs→ GSKit→ Ikeyman Gui Shortcut).** The **IBM Key Management** window is displayed.
2. Click **Key Database File→Open**. The Open window is displayed.
3. Select the key database file in which you want to change the password and click **Open**. The **Password Prompt** window is displayed.
4. Type the password and click **OK**. The **IBM Key Management** window is displayed. The title bar shows the name of the key database file you selected, indicating that the file is open and ready.
5. Click **Key Database File→ Change Password**. The **Change Password** window is displayed.
6. Type a new password in the **Password** field, and type it again in the **Confirm Password** field.
7. Click **OK**. A message in the status bar indicates that the request completed successfully.

Managing digital certificates on a smart card

The iKeyman utility allows you to manage digital certificates on a smart card (or more generically, on a PKCS11 cryptographic token). To do so, you must first perform the following steps to inform the iKeyman utility of the name of the module for managing your smart card:

1. Copy the `ikmuser.sample` file that is shipped with the SSL Toolkit (in the classes directory) to a file called `ikmuser.properties` (also in the classes directory).
2. Edit the `ikmuser.properties` file to set the `DEFAULT_CRYPTOGRAPHIC_MODULE` property to the name of the module for managing your smart card. For example:

   ```
   DEFAULT_CRYPTOGRAPHIC_MODULE=C:\\Winnt\\System32\
   \W32pk2ig.dll
   ```

   The module is normally installed on your system when you install the software for your smart card.
3. Save the `ikmuser.properties` file.

The above configuration steps need only be performed once. Every time the iKeyman utility is started, it will read the contents of your ikmuser.properties file. If you place your smart card in the smart card reader and then start the iKeyman utility, the **IBM Key Management** window will have an additional menu item, called **Cryptographic Token**, as shown in Figure 12 on page 23:

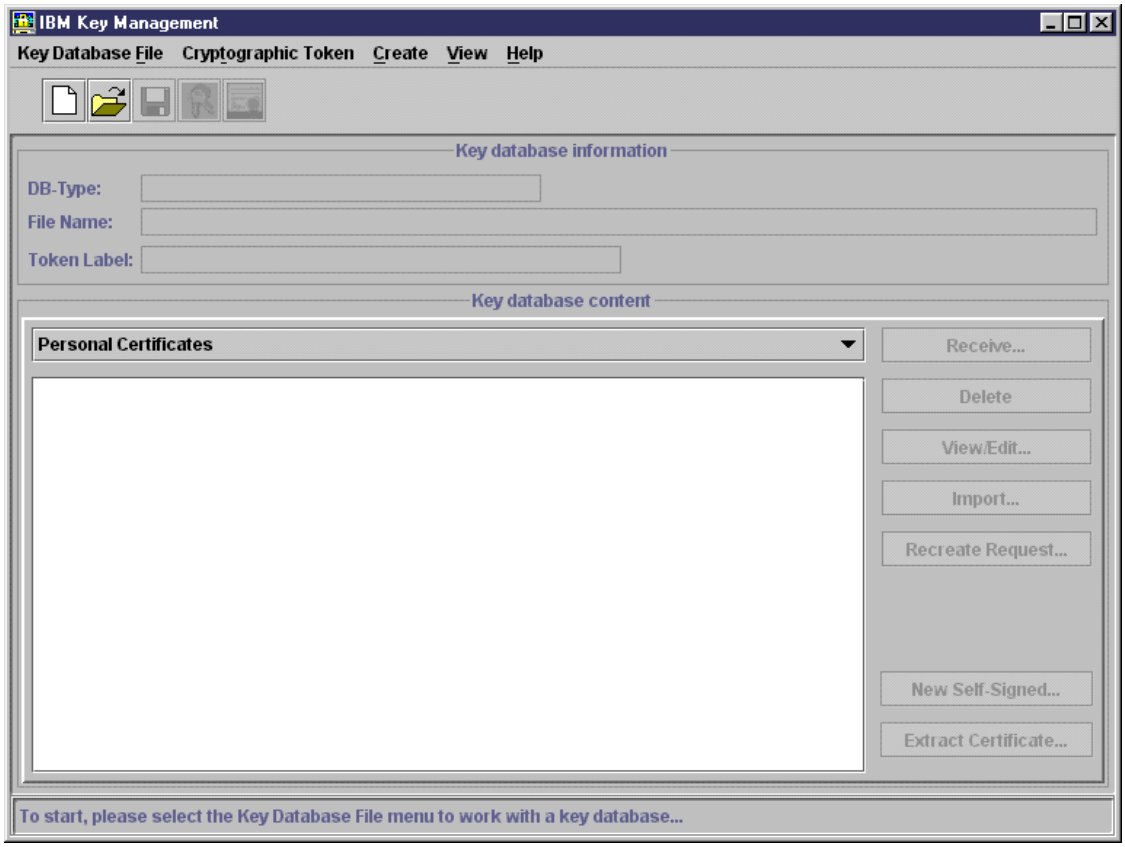

Figure 12. Cryptographic Token in **IBM Key Management** *Window*

To manage digital certificates on your smart card, click **Cryptographic Token→Open**. The **Open Cryptographic Token** window is displayed, as shown in Figure 13 on page 24

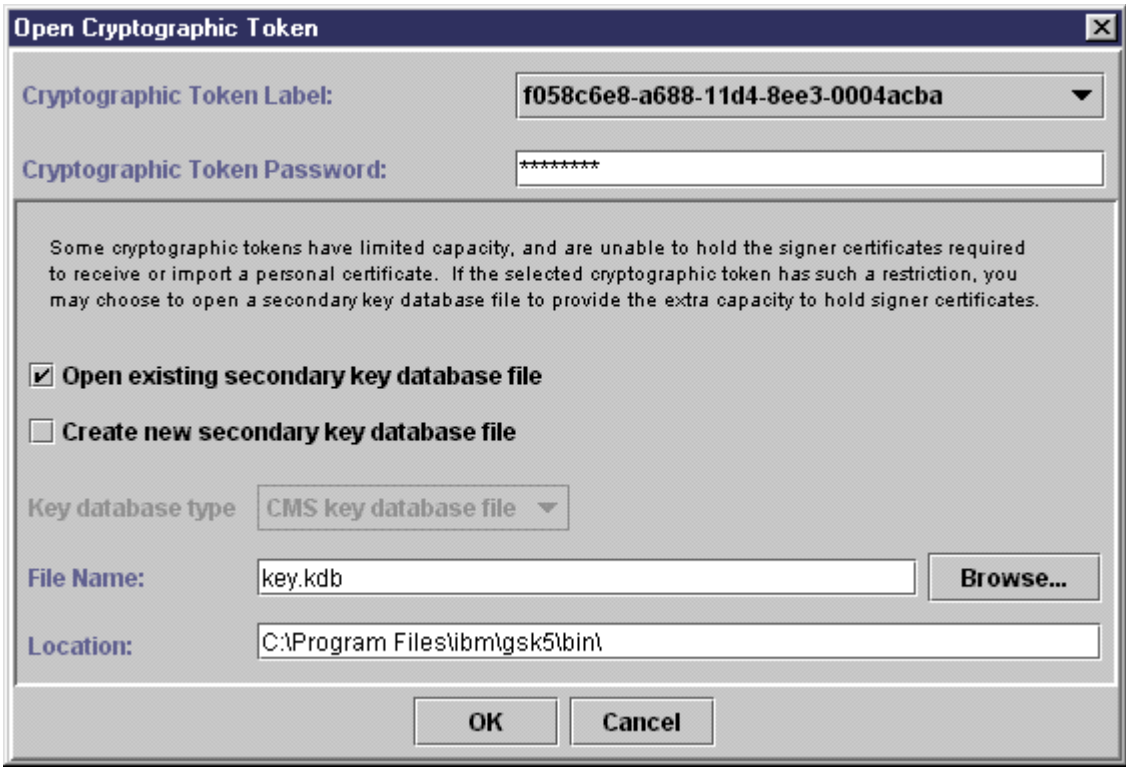
Open Cryptographic Token
Cryptographic Token Label:
f058c6e8-a688-11d4-8ee3-0004acba
Cryptographic Token Password:
****
Some cryptographic tokens have limited capacity, and are unable to hold the signer certificates required to receive or import a personal certificate. If the selected cryptographic token has such a restriction, you may choose to open a secondary key database file to provide the extra capacity to hold signer certificates.
Open existing secondary key database file
Create new secondary key database file
Key database type
CMS key database file
File Name:
key.kdb
Browse...
Location:
C:\Program Files\ibm\gsk5\bin\
OK
Cancel

Figure 13. Open Cryptographic Token Window

Some smart cards have limited capacity, and are unable to hold the signer certificates required to receive or import a personal certificate. If this restriction applies to your smart card, you may open a secondary key database, in addition to the smart card. The secondary key database serves as a container for the signer certificates associated with receiving and importing personal certificates onto your smart card. If your smart card has sufficient capacity, you may not need to open a secondary key database.

To request a digital certificate for a smart card, follow the same steps as those described for requesting a digital certificate for a key database, except that instead of clicking **Key Database->Open**, click **Cryptographic Token->Open**. After specifying the password for the smart card, proceed with the steps as if you had opened a key database.

After the CA sends you a new digital certificate, you need to add it to the smart card from which you generated the request. Follow the same steps as those described for receiving a digital certificate for a key database, except that instead of clicking **Key Database->Open**, click **Cryptographic Token->Open**. It is at this point that you may need to open a secondary key database, if your smart card does not have the capacity to hold the signer certificate(s) associated with the request. Once you have opened the cryptographic token and (optionally) a secondary key database, proceed with the steps as if you had opened a key database.

3

Using the IKEYCMD Command Line Interface

IKEYCMD is a tool, in addition to iKeyman, that can be used to manage keys, certificates and certyificate requests. It is functionally similer to iKeyman and is ment to be run from the command line without a graphical interface. It can be called from native shell scripts and programs to be used when applications prefer to add custom interfaces to certificate and key management tasks. It can create key database files for all of the types that iKeyman currently supports. It can create certificate requests, import CA signed certificates and manage self-signed certificates. It is java based and supported on all of the platforms that the SSL toolkit can be used on.

Use IKEYCMD for configuration tasks related to public-private key creation and management. You cannot use IKEYCMD for configuration options that update the server configuration file, *httpd.conf*. For options that update the server configuration file, you must use the IBM Administration Server.

IKEYCMD uses Java and native command line invocation, enabling IKEYMAN task scripting.

Environment Set Up for IKEYCMD Command Line Interface

To run IKEYMAN , set up environment variables to use the IKEYCMD command line interface as follows:

1. Set your PATH to where your Java or JRE executable resides:

   ```
   EXPORT PATH=/opt/IBMJava/bin:$PATH
   ```

2. Set the following CLASSPATH environment variable:

   ```
   EXPORT CLASSPATH=/usr/local/ibm/gsk/classes/cfwk.zip:/
   usr/local/ ibm/gsk/classes/gsk4cls.jar:$CLASSPATH
   ```

Once completed, IKEYCMD should run from any directory. To run an IKEYCMD command, use the following syntax:

```
java com.ibm.gsk.ikeyman.ikeycmd <command>
```

Note: You can substitute JRE for Java, depending on whether you are using a JRE or JDK.
Example:
`jre com.ibm.gsk.ikeyman.ikeycmd <command>`

IKEYCMD Command Line Syntax

The syntax of the Java CLI is:

```
java [-Dikeycmd.properties=<properties_file>]
com.ibm.gsk.ikeyman.ikeycmd <object> <action> [options]
```

where:	
-Dikeycmd.properties	Specifies the name of an optional properties file to use for this Java invocation. A default properties file, *ikeycmd.properties*, is provided as a sample file that can be modified and used by any Java application.

Object is one of the following:	
-keydb	Actions taken on the key database (either a CMS key database file, a WebDB keyring file, or SSLight class)
-cert	Actions taken on a certificate
-certreq	Actions taken on a certificate request
-help	Display help for the IKEYCMD invocations
-version	Display version information for IKEYCMD

Action is the specific action to be taken on the object, and *options* are the options, both required and optional, specified for the object and action pair.

Note: The *object* and *action* keywords are positional and must be specified in the selected order. However, options are not positional and can be specified in any order, provided that they are specified as an option and operand pair.

User Interface Task Reference

IKEYCMD command line interface tasks are summarized in the following table.

IKEYMAN and IKEYCMD task	For instructions, go to:
Create a new key database and specify the database password	“Creating a new key database” on page 27
Create a new key pair and certificate request	“Creating a new key pair and certificate request” on page 28
Create a self-signed certificate	“Creating a self-signed certificate” on page 29
Export a key to another database or PKCS12 file	“Exporting keys” on page 29
Import a key from another database or PKCS12 file	“Importing keys” on page 29
List certificate authorities (CAs) and certificate requests	“Listing CAs” on page 30
Open a key database	“Opening a key database” on page 30

IKEYMAN and IKEYCMD task	For instructions, go to:
Receive a CA-signed certificate into a key database	“Receiving a CA-signed certificate” on page 30
Show the default key in a key database	“Showing the default key in a key database” on page 31
Store the root certificate of a CA	“Storing a CA certificate” on page 31
Store the encrypted database password in a stash file	“Storing the encrypted database password in a stash file” on page 31

Creating a new key database

A key database is a file that the server uses to store one or more key pairs and certificates. You can use one key database for all your key pairs and certificates, or create multiple databases.

To create a new key database using the IKEYCMD command line interface, enter the following command:

```
java com.ibm.gsk.ikeyman.ikeycmd -keydb -create -db
<filename>.kdb -pw <password> -type cms -expire <days> -stash
```

where:

<pasword> : Password is required for each key database operation. Even though a database of the type sslight requires a specified password,the password can be a NULL string (specified as "").
-type: IBM HTTP Server only handles a cms key database.
-expire: Days before password expires.
-stash: Stashes password for key database. Stashing the password is required for the IBM HTTP Server.

When the -stash option is specified during the key database creation, the password is stashed in a file with a filename built as follows:

```
<filename of key database>.sth
```

For example, if the database being created is named *keydb.kdb*, the stash filename is *keydb.sth*.

Setting the database password

When you create a new key database, you specify a key database password. This password protects the private key. The private key is the only key that can sign documents or decrypt messages encrypted with the public key. Changing the key database password frequently is a good practice.

Use the following guidelines when specifying the password:

- The password must be from the U.S. English character set.
- The password should be at least six characters and contain at least two nonconsecutive numbers. Make sure the password does not consist of publicly obtainable information about you, such as the initials and birth date for you, your spouse, or children.
- Stash the password.

Note: Keep track of expiration dates for the password. If the password expires, a message is written to the error log. The server will start, but there will not be a secure network connection if the password has expired.

Changing the database password

To change the database password, type:

```
java com.ibm.gsk.ikeyman.ikeycmd -keydb -changepw -db <filename>
.kdb -pw <password> -new_pw<new_password> -expire <days> -stash
```

where:
-new_pw: New key database password; this password must be different than the old password and this password cannot be a NULL string.
-expire: Days before password expires.
-stash: Stashes password for key database. Stashing the password is required for the IBM HTTP Server.

Registering a key database with the server

The initial configuration setting for the default key database name is *key.kdb*. If you use *key.kdb* as your default key database name, you do not need to register the database with the server. The server will use the initial setting on the KeyFile directive in the configuration file. If you do not use *key.kdb* as your default key database name, or, if you create additional key databases, you must register those databases.

Creating a new key pair and certificate request

To create a public-private key pair and certificate request:

1. Enter the following command:
   ```
   java com.ibm.gsk.ikeyman.ikeycmd -certreq -create -db
   <db_name>.kdb -pw <password> -size <1024 | 512> -
   dn<distinguished_name>-file <filename> -label <label>
   ```

   where:
   -size: Key size of 512 or 1024
   -label: Label attached to certificate or certificate request
   -dn: X.500 distinguished name. This is input as a quoted string of the following format (Only CN, O, and C are required) CN=common_name, O=organization, OU=organization_unit, L=location, ST=state/province, C=country.
   Example:
   ```
   "CN=weblinux.raleigh.ibm.com,O=ibm,OU=IBM HTTP
   Server,L=RTP,ST=NC,C=US"
   ```
   -file: Name of file where the certificate request will be stored.
2. Verify that the certificate was successfully created.
   a. View the contents of the certificate request file you created.
   b. Make sure the key database recorded the certificate request:
      ```
      java com.ibm.gsk.ikeyman.ikeycmd -certreq -list -db
      <filename>-pw <password>
      ```
      You should see the label listed that you just created.
3. Send the newly created file to a certificate authority.

Creating a self-signed certificate

It usually takes two to three weeks to get a certificate from a well-known CA. While waiting for an issued certificate, use IKEYMAN to create a self-signed server certificate to enable SSL sessions between clients and the server. Use this procedure if you are acting as your own CA for a private Web network.

To create a self-signed certificate, enter the following command:

```
java com.ibm.gsk.ikeyman.ikeycmd -cert -create -db <db_name>.kdb
-pw <password>-size <1024 | 512> -dn<distinguished name> -label
<label> -default_cert <yes or no>
```

where:
-size: Key size 512 or 1024
-label: Enter a descriptive comment used to identify the key and certificate in the database.
-dn: Enter an X.500 distinguished name. This is input as a quoted string of the following format (Only CN, O, and C are required): CN=common_name, O=organization, OU=organization_unit, L=location, ST=state, province, C=country

Example:

```
"CN=weblinux.raleigh.ibm.com,O=ibm,OU=IBM HTTP
Server,L=RTP,ST=NC,C=US"
```

-default_cert: Enter yes, if you want this certificate to be the default certificate in the key database. Enter no, if not.

Exporting keys

To export keys to another key database or to export keys to a PKCS12 file, enter the following command:

```
java com.ibm.gsk.ikeyman.ikeycmd -cert -export -db <filename> -pw
<password> -label <label>-type <cms | sslight> -target <filename>
-target_pw <password> -target_type <cms | sslight | pkcs12>-
encryption <strong | weak>
```

where:
-label : Label attached to the certificate.
-target : Destination file or database.
-target_pw : Password for the target key database.
-target_type : Type of the database specified by -target operand
-encryption : Strength of encryption. Default is strong.

Importing keys

To import keys from another key database, enter the following command:

```
java com.ibm.gsk.ikeyman.ikeycmd -cert -import -db <filename> -
pw <password> -label <label> -type <cms | sslight> -target
<filename> -target_pw <password> -target_type <cms | sslight>
```

To import keys from a PKCS12 file,enter the following command:

```
java com.ibm.gsk.ikeyman.ikeycmd -cert -import -file <filename> -
pw <password> -type pkcs12 -target <filename> -target_pw
<password> -target_type <cms | sslight>
```

where:
-label : Label attached to the certificate.
-target : Destination database.
-target_pw : Password for the key database if -target specifies a key database
-target_type : Type of the database specified by -target operand.

Listing CAs

To display a list of trusted CAs in a key database:

```
java com.ibm.gsk.ikeyman.ikeycmd -cert -list CA -db <dbname>.kdb
-pw <password>-type cms
```

Opening a key database

There is no explicit opening of a key database. For each command, database and password options are specified and these specifications provide the information needed to operate in a key database.

Receiving a CA-signed certificate

Use this procedure to receive an electronically mailed certificate from a certificate authority (CA), designated as a trusted CA on your server. By default, the following CA certificates are stored in the key database and marked as trusted CA certificates:

- IBM World Registry CA
- Integrion CA Root (from IBM World Registry)
- VeriSign Class 1 Public Primary CA
- VeriSign Class 2 Public Primary CA
- VeriSign Class 3 Public Primary CA
- VeriSign Class 4 Public Primary CA
- VeriSign Test CA
- RSA Secure Server CA (from VeriSign)
- Thawte Personal Basic CA
- Thawte Personal Freemail CA
- Thawte Personal Premium CA
- Thawte Premium Server CA
- Thawte Server CA

The Certificate Authority may send more than one certificate. In addition to the certificate for your server, the CA may also send additional Signing certificates or Intermediate CA Certificates. For example, Verisign includes an Intermediate CA Certificate when sending a Global Server ID certificate. Before receiving the server certificate, receive any additional Intermediate CA certificates. Follow the instructions in “Storing a CA certificate” on page 31 to receive Intermediate CA Certificates.

Note: If the CA who issues your CA-signed certificate is not a trusted CA in the key database, you must first store the CA certificate and designate the CA as a trusted CA. Then you can receive your CA-signed certificate into the database. You cannot receive a CA-signed certificate from a CA who is not a trusted CA. For instructions, see "Storing a CA certificate".

To receive the CA-signed certificate into a key database, enter the following command:

```
java com.ibm.gsk.ikeyman.ikeycmd -cert -receive -file <filename>
-db <db_name>.kdb -pw <password> -format <ascii | binary> -
default_cert <yes | no>
```

where:
-format: Certificate Authority might provide CA Certificate in either ascii or binary format
-label: Label attached to CA certificate.
-trust: Indicates whether this CA can be trusted. Use enable options when receiving a CA certificate.
-file: File containing the CA certificate.

Showing the default key in a key database

To display the default key entry:

```
java com.ibm.gsk.ikeyman.ikeycmd -cert -getdefault -db
<dbname>.kdb -pw <password>
```

Storing a CA certificate

To store a certificate from a CA who is not a trusted CA:

```
java com.ibm.gsk.ikeyman.ikeycmd -cert -add -db <filename>.kdb -
pw <password>-label <label> -format <ascii | binary> -trust
<enable |disable> -file<file>
```

where:
-label: Label attached to certificate or certificate request
-format: Certificate Authorities might supply a binary ASCII file
-trust: Indicate whether this CA can be trusted. Should be Yes.

Storing the encrypted database password in a stash file

For a secure network connection, store the encrypted database password in a stash file.

To store the password while a database is created:

```
java com.ibm.gsk.ikeyman.ikeycmd -keydb -create -db
<path_to_db>/<db_name>.kdb-pw <password> -type cms -expire <days>
-stash
```

To store the password after a database has been created:

```
java com.ibm.gsk.ikeyman.ikeycmd -keydb -stashpw -db
<db_name>.kdb -pw <password>
```

Using GSK5CMD batch file

A batch file, gsk5cmd , provide the same function of "java com.ibm.gsk.ikeyman" command. For example, to store the password after a database has been created, you can also enter following command :

```
gsk5cmd -keydb -stashpw -db <db_name>.kdb -pw <password>
```

IKEYCMD Command Line Parameter Overview

The following table describes each *action* that can be performed on a specified *object*.

Object	Action	Description
-keydb	-changepw	Change the password for a key database
	-convert	Convert the key database from one format to another
	-create	Create a key database
	-delete	Delete the key database
	-stashpw	Stash the password of a key database into a file
-cert	-add	Add a CA certificate from a file into a key database
	-create	Create a self-signed certificate
	-delete	Delete a CA certificate
	details	List the detailed information for a specific certificate
	-export	Export a personal certificate and its associated private key from a key database into a PKCS#12 file, or to another key database
	-extract	Extract a certificate from a key database
	-getdefault	Get the default personal certificate
	-import	Import a certificate from a key database or PKCS#12 file
	-list	List all certificates
	-modify	Modify a certificate (NOTE: Currently, the only field that can be modified is the Certificate Trust field)
	-receive	Receive a certificate from a file into a key database
	-setdefault	Set the default personal certificate
	-sign	Sign a certificate stored in a file with a certificate stored in a key database and store the resulting signed certificate in a file
-certreg	-create	Create a certificate request
	-delete	Delete a certificate request from a certificate request database
	-details	List the detailed information of a specific certificate request
	extract	Extract a certificate request from a certificate request database into a file

Object	Action	Description
	-list	List all certificate requsts in the certificate request database
	-recreate	Recreate a certificate request
-help		Display help information for the IKEYCMD command
-version		Display IKEYCMD version information

IKEYCMD Command Line Options Overview

The following table shows each option that can be present on the command line. The options are listed as a complete group. However, their use is dependent on the *object* and *action* specified on the command line.

Option	Description
-db	Fully qualified path name of a key database.
-default_cert	Sets a certificate to be used as the default certificate for client authentication (yes or no). Default is no.
-dn	X.500 distinguished name. Input as a quoted string of the following format (only CN, O, and C are required): *"CN=Jane Doe,O=IBM,OU=Java Development,L=Endicott, ST=NY,ZIP=13760,C=country"*
-encription	Strength of encryption used in certificate export command (strong or weak). Default is strong.
-expire	Expiration time of either a certificate or a database password (in days). Defaults are: 365 days for a certificate and 60 days for a database password.
-file	File name of a certificate or certificate request (depending on specified *object*).
-format	Format of a certificate (either ascii for Base64_encoded ASCII or *binary* for Binary DER data). Default is ascii.
-label	Label attached to a certificate or certificate request.
-new_format	New format of key database.
-new_pw	New database password.
-old_format	Old format of key database.
-pw	Password for the key database or PKCS#12 file. See “Creating a new key database” on page 27.
-size	Key size (512 or 1024). Default is 1024.
-stash	Indicator to stash the key database password to a file. If specified, the password will be stashed in a file.
-target	Destination file or database.
-target_pw	Password for the key database if *-target* specifies a key database. See “Creating a new key database” on page 27.
-target_type	Type of database specified by *-target* operand (see *-type*).

Option	Description
-trust	Trust status of a CA certificate (enable or disable). Default is enable.
-type	Type of database. Allowable values are cms (indicates a CMS key database), webdb (indicates a keyring), sslight (indicates an SSLight .class), or pkcs12 (indicates a PKCS#12 file).
-x509version	Version of X.509 certificate to create (1, 2 or 3). Default is 3.

Command Line Invocation

The following is a list of each of the command line invocations, with the optional parameters specified in *italics*.

Note: For simplicity, the actual Java invocation, *java com.ibm.gsk.ikeyman,iKeycmd*, is omitted from each of the command invocations.

```
-keydb -changepw -db <filename> -pw <password> -new_pw
<new_password> -stash -expire <days>

-keydb -convert -db <filename> -pw <password> -old_format <cms |
webdb>-new_format <cms>

-keydb -create -db <filename> -pw <password> -type <cms | sslight>
-expire <days> -stash

-keydb -delete -db <filename> -pw <password>

-keydb -stashpw -db <filename> -pw <password>

-cert -add -db <filename> -pw <password> -label <label> -file
<filename> -format <ascii | binary> -trust <enable | disable>

-cert -create -db <filename> -pw <password> -label <label> -dn
<distinguished_name> -size <1024 | 512> -x509version <3  | 1 | 2>
-default_cert <no | yes>

-cert -delete -db <filename> -pw <password> -label <label>

-cert -details -db <filename> -pw <password> -label <label>

-cert -export -db <filename> -pw <password> -label <label> -type
<cms | sslight> -target <filename> -target_pw <password>
-target_type <cms | sslight | pkcs12> -encryption <strong | weak>

-cert -extract -db <filename> -pw <password> -label <label> -
target <filename> -format <ascii | binary>
```

```
-cert -getdefault -db <filename> -pw <password>

-cert -import -db <filename> -pw <password> -label <label> -type
<cms | sslight> -target <filename> -target_pw <password> -
target_type <cms | sslight>

-cert -import -file <filename> -type <pkcs12> -target <filename>
-target_pw <password> -target_type <cms | sslight>

-cert -list <all | personal | CA | site> -db <filename> -pw
<password> -type <cms | sslight>

-cert -modify -db <filename> -pw <password> -label <label>  -trust
<enable | disable>

-cert -receive -file <filename> -db <filename> -pw <password>
-format <ascii | binary> -default _cert <no | yes>

-cert -setdefault -db <filename> -pw <password> -label <label>

-cert -sign -file <filename> -db <filename> -pw <password> -label
<label> -target <filename> -format <ascii | binary>  -expire
<days>

-certreq -create -db <filename> -pw <password> -label <label> -dn
<distinguished_name> -size <1024 | 512> -file <filename>

-certreq -delete -db <filename> -pw <password> -label <label>

-certreq -details -db <filename> -pw <password> -label <label>

-certreq -extract -db <filename> -pw <password> -label <label> -
target <filename>

-certreq -list -db <filename> -pw <password>

-certreq -recreate -db <filename> -pw <password> -label <label> -
target <filename>

-help

-version
```

User Properties File

In order to eliminate some of the typing on the Java CLI invoacations, user properties can be specified in a properties file. The properties file can be specified on the Java command line invocation via the *-Dikeycmd.properties* Java option. A sample properties file, *ikeycmd.properties*, is supplied as a sample to enable Java applications to modify default settings for their application.

Error Messages

TBD

Index

A

adding a CA root digital certificate 15
asymmetric cryptography 1
authentication
  client 5, 10

C

CA root digital certificate
  adding 15
  deleting 15
certificate authority (CA)
  and digital certificates 2
  and trust hierarchies 3
  root CA 4
chain of trust 3, 4, 9
changing a database password 22
client authentication 5, 10
creating a key database 11
creating a self-signed digital certificate 14
cryptography
  asymmetric 1
  description 1
  export of 7
  public-key 1

D

database
  changing a password 22
  creating a key database 11
default key, setting 20
deleting a CA root digital certificate 15
deleting a digital certificate 20
digital certificate
  adding a root CA 15
  and SSL and trust chains 9
  certificate authority (CA) 2, 3
  deleting 20
  deleting a root CA 15
  expiration 2
  extracting 16
  format 2
  global server certificate 7
  layout 2
  overview 1
  purpose 2
  receiving 19
  requesting 6, 18
  RSA Secure Server CA 13
  security considerations 3
  self-signed 4, 6, 14
  setting a new default key 20
  signed 6
  signer 13
  Thawte Personal Basic CA 13
  Thawte Personal Freemail CA 13
  Thawte Personal Premium CA 13
  Thawte Premium Server CA 13
  Thawte Server CA 13
  uses 5
  VeriSign Class 1 Public Primary CA 13
  VeriSign Class 2 Public Primary CA 13
  VeriSign Class 3 Public Primary CA 13
  VeriSign Test CA Root Certificate 13, 14
digital signature, issuer's 2
distinguished name
  issuer's 2
  owner's 2
double handshake 10

E

electronic mail 5
encryption 1
export of cryptographic hardware and software 7
extracting a digital certificate 16

F

format of digital certificate 2

G

global server certificate
  description 7
  with SSL 10

H

handshake
  description 8
  double 10
hierarchy of trust 3

HTTP 7

I

IKE standard 5
iKeyman
  adding a CA root digital certificate 15
  changing a database password 22
  creating a key database 11
  creating a self-signed digital certificate 14
  deleting a CA root digital certificate 15
  deleting a digital certificate 20
  extracting a digital certificate 16
  receiving a digital certificate 19
  requesting a digital certificate 18
  setting a new default key 20
IP Security 5
IPsec standard 5
issuer's distinguished name 2

K

key database, creating 11
key pair 1
key ring 9

L

layout of digital certificate 2
Lightweight Directory Access Protocol (LDAP) 5

M

master secret 9
message authentication code (MAC) 9

O

owner's distinguished name 2
owner's public key 2

P

password, changing for a database 22
PEM 5
pre-master secret 9
private key 1, 3
private-public key pair 3
public key 1
public key infrastructure (PKI) 9
public key, owner's 2
public-key cryptography 1
public-private key pair 1, 3

R

receiving a digital certificate 19
requesting a digital certificate 6, 18
root CA 4
RSA Secure Server CA 13

S

S/MIME 5
secure electronic mail 5
secure electronic transaction (SET) 5
secure tunnels 5
security considerations 3
self-signed digital certificate
  contents 6
  creating 14
  description 4
SET 5
setting a new default key 20
signed digital certificate 6
signer digital certificates, list of 13
SSL
  and digital certificates and trust chains 9
  definition 5
  double handshake 10
  encryption 1
  handshake 8
  how SSL works 7
  overview 1, 7
  with global server certificates 10
SSL step-up protocol 10
step-up protocol 10

T

Thawte
  Personal Basic CA 13
  Personal Freemail CA 13
  Personal Premium CA 13
  Premium Server CA 13
  Server CA 13
trust chain
  and digital certificates and SSL 9
  description 3, 4

trust hierarchy 3

V

VeriSign, Inc.
  authorization from US government 7
  Class 1 Public Primary CA 13
  Class 2 Public Primary CA 13
  Class 3 Public Primary CA 13
  Test CA Root Certificate 13, 14
virtual private network (VPN) 5

Part Number:

File Number:

Printed in the United States of America
on recycled paper containing 10&
recovered post-consumer fiber.

0000-0000-00

Tivoli